VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV MIKROELEKTRONIKY
DEPARTMENT OF MICROELECTRONICS

ELEKTRONICKÝ DOCHÁZKOVÝ SYSTÉM
ELECTRONIC ATTENDANCE SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Adam Valent**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Josef Šandera, Ph.D.**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Mikroelektronika a technologie**
Ústav mikroelektroniky

Student: Adam Valent ***ID:*** 195458
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Elektronický docházkový systém

POKYNY PRO VYPRACOVÁNÍ:

Navrhněte a realizujte elektronický docházkový systém. V systému budou použity karty RFID. Data se budou zapisovat na HTTP server přes zabezpečené připojení a zároveň zálohovat na kartu SD. Docházkový systém bude obsahovat zobrazovací jednotku, na které se budou zobrazovat aktuální informace.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce.

Termín zadání: 4.2.2019 ***Termín odevzdání:*** 30.5.2019

Vedoucí práce: doc. Ing. Josef Šandera, Ph.D.
Konzultant:

doc. Ing. Jiří Háze, Ph.D.
předseda oborové rady

Abstrakt

Cieľom bakalárskej práce „Elektronický dochádzkový systém“ je vyrobiť plnohodnotný interaktívny a moderný dochádzkový systém s web stránkou, serverom a databázou. Zamestnanci budú vytvárať záznamy o dochádzke priložením RFID čipu k hardvérovej časti dochádzkového systému. Záznamy z databázy bude možné prehliadať, upravovať a exportovať do CSV súboru.

Kľúčové slová

dochádzkový systém, server, databáza, mikrokontrolér, web

Abstract

The goal of the bachelor thesis “Electronic Attendance System” is create a full-featured interactive and modern attendance system with web page, server and database. Employees will be able to create attendance records by attaching the RFID chip to hardware part of an attendance system. Database records will be possible to browse, edit end export to a CSV file.

Keywords

attendance system, server, database, microcontroller, web

Bibliografická citácia

VALENT, Adam. *Elektronický docházkový systém.* Brno, 2019. Dostupný taktiež z: https://www.vutbr.cz/studenti/zav-prace/detail/119448. Bakalárska práca. Vysoké učení technické v Brne, Fakulta elektrotechniky a komunikačných technológií, Ústav mikroelektroniky. Vedúci práce doc. Ing. Josef Šandera, Ph.D.

Poďakovanie

Ďakujem vedúcemu bakalárskej práce, docentovi Ing. Josefovi Šanderovi Ph.D., za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri spracovaní mojej bakalárskej práce. Ďalej by som chcem poďakovať konzultantovi, Ing. Ondrejovi Palkocimu, za korekciu a odborné vedenie pri praktickej časti bakalárskej práce.

V Brne dňa: **30. mája 2019** ……

podpis autora

Obsah

Úvod

Cieľom tejto práce je skonštruovať dochádzkový systém, ktorého výroba bude praktickým prínosom skúseností vo viacerých odborných oblastiach. Fyzická časť dochádzkového systému bude zostavená zo stavebných blokov, tzv. modulov. Všetka elektronika bude v šasi vytlačenej na mieru za pomoci 3D tlačiarne. Moduly budú komunikovať s mikrokontrolérom pomocou zberníc, ten potom bude odosielať údaje o dochádzke cez WiFi na HTTP server a zároveň ich zálohovať na SD kartu. WiFi komunikácia bude prebiehať cez SSL spojenie kvôli bezpečnosti. Server prijaté záznamy o dochádzke uloží do lokálnej databázy. Na stránku serveru sa bude možné prihlásiť sa a po prihlásení si zobraziť vytvorenú dochádzku. Dochádzku bude možné prehliadať, upravovať a sťahovať vo formáte CSV, ktorý je podporovaný Excelom.

Pri práci je dobrým zvykom si napísaný kód zálohovať a ani teraz tomu nebude inak. Kód a všetky jeho predchádzajúce verzie budú zálohované pomocou platformy Git a nahraté na GitHub server. Celý zdrojový kód bude na tejto stránke verejne dostupný. Na nej bude k dispozícii pre prípadné obnovenie pracovnej kópie v počítači, alebo pre distribúciu.

1 Súčasti riešenia

Dochádzkový systém je dnes samozrejmosťou v každej firme, ktorá platí zamestnancov podľa počtu odpracovaných hodín. Ide o zariadenie, ktoré zaznamenáva dochádzku pre výpočet mzdy zamestnanca. Pred niekoľkými desiatkami rokov bola dochádzka realizovaná pomocou štikacích lístkov, ktoré sa označovali v tzv. štikacích hodinách. Bolo to mechanické zariadenie označujúce kartičku časom. V dnešnej dobe sú dochádzkové systémy riešené elektronicky. Zamestnanca dokážu rozoznať pomocou kartičky, otlačku prsta, alebo šošovky. Dochádzka sa ukladá buď na SD kartu alebo sa odosiela do databázy, ktorá je prípadne zálohovaná. Keďže sa z údajov o dochádzke určuje mzda, je nutné, aby tieto údaje boli bezpečne uložené a nestalo sa, že na konci mesiaca niektoré záznamy chýbajú.

Tento dochádzkový systém je zložený z fyzickej časti (šasi s čítačkou kariet, mikrokontrolérom, displejom atď.) a serverovej časti (server s databázou). Fyzická časť ukladá dochádzku na SD kartu a zároveň ju odosiela na server, ten ju následne ukladá do databázy. Z webového prehliadača je možné komunikovať so serverom a dochádzku prezerať, upravovať, alebo ju stiahnuť vo formáte CSV. Jadro fyzickej časti je mikrokontrolér ESP32, ktorý komunikuje s ostatnými modulmi pomocou zberníc UART, I^2C a SPI. LCD displej používa 5 V logiku a mikrokontrolér nemá vstupno-výstupné piny 5 V kompatibilné. Preto je medzi LCD a mikrokontrolérom prevodník logických úrovní, ktorý mení úroveň napätia v oboch smeroch. Jadro serverovej časti je program NodeJS a databáza je typu MongoDB. Databáza beží lokálne na serveri.

1.1 ESP32

ESP32 je mikrokontrolér od firmy Espressif Systems, ktorý je použitý na obsluhu hardvéru, na trhu je od roku 2016, takže ide o relatívne nové zariadenie. Jedná sa o nízko-spotrebný, lacný a veľmi výkonný mikrokontrolér s množstvom periférií a funkcií. Mikroprocesor je vyrobený v 40 nm technológii. Napájací rozsah je od 2,2 V do 3,6 V. Je dôležité poznamenať, že vstupno-výstupné piny nie sú kompatibilné pre 5 V TTL logiku. Privedenie takéhoto napätia by mohlo vstupno-výstupný pin zničiť. Pri komunikácií s 5 V logikou sa pre to používa prevodník logických úrovní. Pre komunikáciu cez WiFi alebo Bluetooth je čip prispájkovaný na malej doske plošného spoja s anténou. Celá doska sa volá ESP32 - wroom - 32 a obsahuje okrem antény aj pár pasívnych súčiastok, kryštál

a Flash pamäť spojenú SPI zbernicou. Všetko okrem antény je zakryté plechovým krytom kvôli elektromagnetickému tieneniu. S množstvom funkcií, ktoré ponúka, ide takmer o univerzálny mikrokontrolér, ktorý vie spracovať video, alebo audio dáta a prenášať ich bezdrôtovo cez Bluetooth, alebo WiFi. Doska má po celom obvode, okrem strany, kde je anténa, polovičné prekovy (z angličtiny castellated mounting holes) s povrchovou úpravou ENIG. Aj keď by táto doska fungovala bez problémov aj sama, je ešte prispájkovaná za polovičné prekovy (ako SMD súčiastka) na ďalšej a väčšej doske s napäťovým regulátorom a radičom pre komunikáciu cez USB. Takto sa dá mikrokontrolér pripojiť k počítaču a naprogramovať cez USB kábel. Názov tejto veľkej dosky je DOIT ESP32 DEVKIT V1.

1.1.1 Funkcie a periférie

Mikrokontrolér je priam nabitý veľkým množstvom funkcií a periférií, ale pre obsluhu dochádzkového systému sú potrebné len niektoré z nich. Disponuje dvojjadrovým procesorom Xtensa so šírkou slova 32 bitov a frekvenciou jadra 80 Mhz. Ďalej sa v ňom nachádza nízko spotrebný koprocesor, 520 KiB pamäte typu SRAM, 448 KiB pamäte typu ROM, bezdrôtová komunikácia Wi-Fi so štandardom 802.11, desať-bitový analógovo-digitálny prevodník, Bluetooth 4.2 a zbernice SPI, I^2C, I^2S, UART a CAN. Plný zoznam periférií sa nachádza v technickom liste.[1]

Pozoruhodná je aj veľkosť mikrokontroléra s tak veľa perifériami. Jeho rozmery sú 6x6mm. Je samozrejmé, že pri niektorých aplikáciách ktoré využijú jeho plný potenciál sa bude značne zahrievať. Preto je v puzdre QFN s veľkou zemniacou plôškou na odvádzanie tepla zospodu. Maximálna prúdová zaťažiteľnosť výstupného pinu je 40 mA. Mikrokontrolér je testovaný na elektrostatický výboj s napätím 1500V, jeden a pol násobné prekročenie napájacieho napätia a 60 % relatívnu vlhkosť pri 30 °C na 192 hodín.

1.2 Zbernice

Zbernica je sústava vodičov na prenášanie informácie. Aby sa informácia preniesla, musia obidve strany dodržiavať komunikačný protokol. Ten určuje koľko vodičov sa má použiť, aké časovanie má byť medzi jednotlivými bitmi a podobne. Existujú sériové a paralelné zbernice. V sériových zberniciach sa prenáša informácia v rade bitov za sebou. Sériové zbernice sú pomalšie ale na prenos stačí aj jeden vodič. Paralelné zbernice prenášajú dáta

kombináciou logických úrovní na jednotlivých vodičoch. Výhoda je, že prenos je rýchlejší, ale je naň potreba viacej vodičov.

Zbernice okrem rozdelenia na paralelne a sériové, môžu byť rozdelené aj na synchrónne a asynchrónne. Synchrónne sú, ak sa riadia hodinovým signálom (časté označenie CLK z anglického slova clock), ktorý im hovorí kedy majú prečítať napätie na komunikačných vodičoch. Asynchrónne nemajú hodinový signál, ale riadia sa presným časovaním a rozostupom bitov na zbernici. Zariadenie pripojené k zbernici môže byť vo funkcií nadradeného zariadenia (master) alebo podradeného zariadenie (slave). Nadradené zariadenie rozhoduje kedy sa budú posielať údaje a väčšinou ide o mikrokontrolér. Podradené zariadenie posiela alebo príma údaje na príkaz nadradeného.

1.2.1 SPI

Skratka SPI je z anglického slovného spojenia Serial Peripheral Interface čo znamená sériové periférne rozhranie. Zbernica dokáže komunikovať s rýchlosťou až do 70 MHz.[8] Ide o zbernicu, ktorá umožňuje komunikáciu s jedným a viac zariadeniami uzlovým spojením komunikačných vodičov. Skladá sa z vodičov SCLK, MOSI, MISO a SS (niekedy pod názvom CS). SCLK rozvádza hodinový signál od nadradeného zariadenia, tak môžu všetky zariadenia odosielať dáta synchrónne. MOSI je na prenos dát od nadradeného zariadenia podradenému. MISO je na prenos dát od podradeného zariadenia nadradenému. SS pin určuje, s ktorým zariadením sa na zbernici komunikuje. Ak ho nadradené zariadenie stiahne na nulový potenciál, znamená to, že ide posielať dáta. Preto má každé podradené zariadenie vlastný SS vodič.

1.2.2 I^2C

I^2C je sériová zbernica ktorá má len dva vodiče SDA a SCL. SCL rozvádza k podradeným zariadeniam hodinový signál od nadradeného a SDA je vodič na obojsmerný prenos dát. V najrýchlejšom móde je táto zbernica schopná komunikovať s rýchlosťou 3,4 MHz.[8] Na rozdiel od SPI táto zbernica používa na selekciu zariadenia jeho špecifickú adresu, ktorú pošle so štart bitom po SDA vodiči. Tak podriadené zariadene vie, že dáta zo sériovej linky partia jemu. Po adrese nasleduje miesto v pamäti, kde sa majú dáta uložiť a potom nasledujú samostatné dáta ukončené stop bitom.

1.2.3 UART

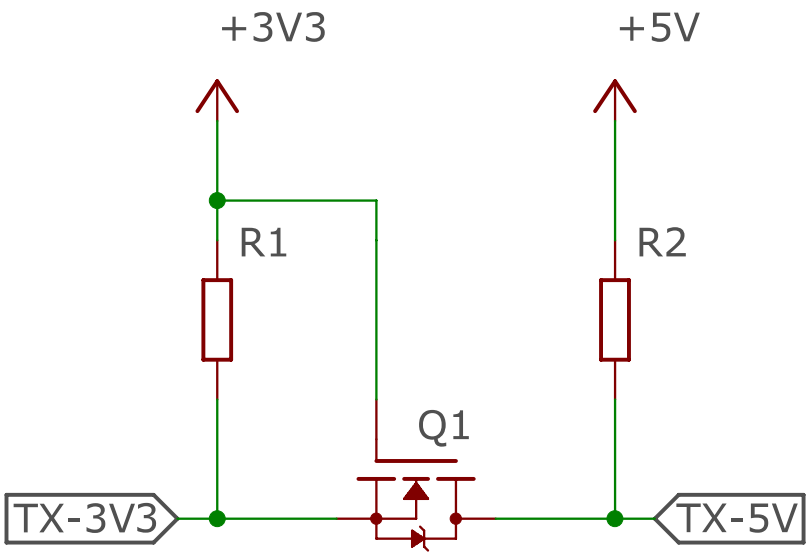
UART komunikácia sa líši od dvoch prvých tým, že nemá hodinový signál, ale prenos prebieha presným časovaním. Táto komunikačná linka dokáže prenášať dáta iba medzi dvoma zariadeniami. Výstupné svorky sa volajú Rx a Tx. Rx značí príjem a Tx odosielanie dát. Rx prvého zariadenie sa spojí s Tx druhého zariadenia. Ak je potrebný iba jednosmerný prenos, stačí na túto komunikáciu iba jeden vodič, čo je jej hlavná výhoda.

1.3 Prevodník logických úrovní

V dnešnej dobe sa čoraz viac čipov navrhuje na napätie 3,3 V. Je pre to viac dôvodov a to, že pri nižšom napätí je aj nižšia spotreba, čipy s veľkým výkonom sa menej zahrievajú a pri nižšom napätí sa v parazitných kapacitách uloží menej náboja, takže čip môže pracovať vo vyšších frekvenciách. Nie však každý 3,3 V čip používa zbernicu tolerantnú na 5 V. Preto ak majú spolu dva zariadenia s rôznym napätím komunikovať, je nutné použiť prevodník logických úrovní. Je to zapojenie, ktoré z vysokého napätia (5 V) spraví nízke (3 V) a naopak. Funguje v oboch smeroch. Jeho schéma je na obrázku 1.3.1-1.

1.3.1 Princíp činnosti

Toto zapojenie môže byť pri používaní v štyroch rôznych stavoch. Prvé dva stavy sú, ak je 3,3 V strana ako vstup pre komunikáciu, kde môže signál nadobúdať 0 alebo 3,3 V. Ak je na 3,3 V vstupe 0 V, potenciál medzi gate a source tranzistora Q1 bude 3,3 V, takže tranzistor sa otvorí a 0 V na 3,3 V strane sa prenesie na 5 V stranu. Ak potom prejde signál do vysokej úrovne, tranzistor sa zavrie a rezistor R2 vytiahne napätie na druhej strane na 5 V. Druhé dva stavy sú, ak je 5 V strana ako vstup, kde signál môže nadobúdať hodnoty 0 alebo 5 V. Ak bude na 5 V strane 0 V, vytvorí sa delič napätia zložený z rezistoru R1 a diódy tranzistora Q1. Takže napätie na druhej strane bude napätie na dióde tranzistora. Pre 3,3 V logiku je nízka úroveň do 0,8 V, preto je potrebné, aby tranzistor mal v sebe diódu s čo najmenším úbytkom napätia. Ak potom prejde signál do vysokej úrovne dióda sa uzavrie a na druhej strane vytiahne rezistor R1 napätie na 3,3 V. [9]



1.3.1-1 schéma zapojenia prevodníka

1.4 NodeJS

NodeJS je open-source multiplatformové prostredie pre spúšťanie JavaScript kódu mimo prehliadača.[3] Pred tým, ako existoval NodeJS JavaScript, kód bežal iba v prehliadačoch. Každý prehliadač má JavaScript engine, ktorý preloží JavaScript kód do strojového kódu. Napríklad Microsoft Edge používa JavaScript engine s názvom Chakra, Mozila Firefox má SpiderMonkey a Google Chrome používa V8. Každý prehliadač používa vlastnú verziu enginu, čo môže spôsobiť že rovnaký JavaScript kód sa bude v inom prehliadači správať trochu inak. V roku 2009 Ryan Dahl zobral Chrome V8 engine, ktorý je najrýchlejší zo všetkých, vstaval ho do C++ programu a nazval ho NodeJS.[2] NodeJS sa ale úplne nezhoduje s Chrome V8. Napríklad v NodeJS sú moduly, ktoré pracujú s priečinkami alebo s HTTP protokolom. NodeJS sa hlavne používa na back-end služby pre web aplikácie, alebo mobilné aplikácie. Tieto aplikácie sú to, čo používateľ vidí a s čím interaguje a pracuje. Je to len povrch celého systému a pre funkciu potrebujú komunikovať s programom bežiacim na serveri alebo cloude, akým je aj program NodeJS. Takto môžu aplikácie ukladať dáta, posielať emaily, dostávať upozornenia, atď. Výhoda tohto prostredia je, že kompiluje JavaScript, čo je vysokoúrovňový programovací jazyk, takže programátor sa vie zamerať na riešenie hlavného problému a nemusí sa zaťažovať napríklad kopírovaním textového reťazca, alebo správnym určením dátových typov, alebo ich pretypovaním. Navyše kód na web stránkach je tiež v JavaScripte takže to veľmi uľahčuje prácu, keďže stránky aj server sú písané v rovnakom programovacom jazyku.

Funkcie v NodeJS aj knižnice z npm (bude vysvetlené neskôr) sú písané asynchrónne (po anglicky non-blocking), čo znamená že ho nezdržujú asynchrónne podnety v hlavnej časti programu. NodeJS pokračuje vo svojom programe a začne ich spracovávať až vtedy keď sa vráti návratová hodnota z ich funkcie.[4]

```
//synchrónny kód
var getUserSync = require(‘./getUserSync.js‘);

var user1 = getUserSync(‘123‘);
console.log(‘user1‘, user1);

var user2 = getUserSync(‘456‘);
console.log(‘user2‘, user2);

var suma = 1 + 2;
console.log(‘Suma je :‘,suma);
```

Synchrónny JavaScript kód čaká, kým sa dokončí predchádzajúca funkcia a až tak pokračuje ďalej.

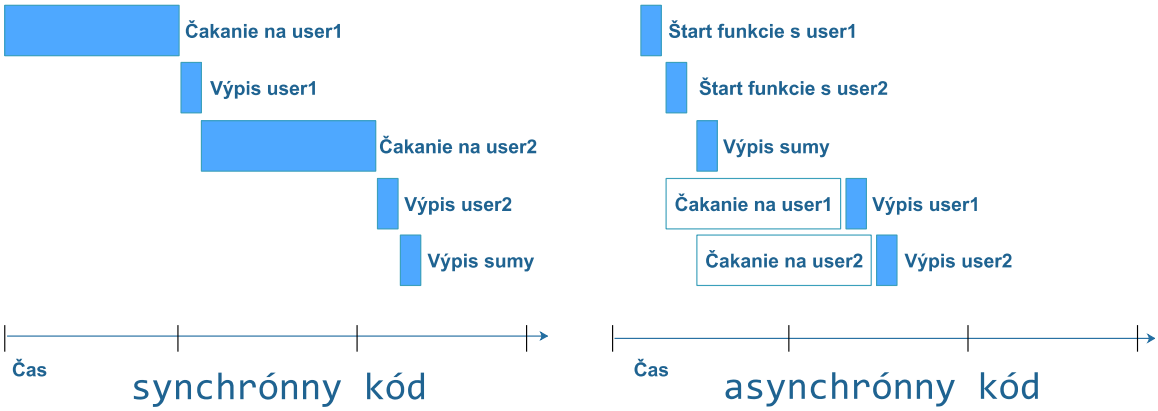
```
//asynchrónny kód
var getUserSync = require(‘./getUserSync.js‘);

getUserSync(‘123‘, function (user1) {
console.log(,‘user1‘, user1);
});

getUserSync(‘456‘, function (user2) {
console.log(,‘user2‘, user2);
});

var suma = 1 + 2;
console.log(‘Suma je :‘,suma);
```

Asynchrónny JavaScript kód prejde až na koniec bez čakania na návratovú hodnotu z asynchrónnych funkcií. Časový rozdiel je znázornený na obrázku 1.4-1. Je teda jasné, že toto je jedna z hlavných výhod NodeJS. Funkcia v tomto prípade ako výpis sumy, ktorá nezávisí na predchádzajúcich funkciách, môže byť vykonaná hneď bez zbytočného čakania. V reálnom prípade ide väčšinou o HTTP požiadavky z užívateľskej časti programu. V NodeJS bežia napríklad PayPal, Uber alebo Netflix.



Obr. 1.3.1-1 porovnanie synchrónneho a asynchrónneho kódu

1.4.1 Node package manager

NodeJS používa najväčšiu verejne dostupnú knižnicu s modulmi, ktoré pokrývajú prácu takmer so všetkým. Jej názov je npm, čo je skratka z anglického spojenia node package manager, ktorá je dostupná online.[14] Na tejto stránke je popis modulov od programátorov, informácie o počte stiahnutí za posledný týždeň, zmeny v nadchádzajúcej verzií a podobne. Sťahovať sa zo stránky ale nedajú. Na to slúži program npm, ktorý ich stiahne, aj nainštaluje. Všetko pomocou jedného príkazu npm install názov modulu. Okrem toho je s npm možné inicializovať súbor, v ktorom sa bude ukladať história všetkých nainštalovaných modulov v projekte. Vytvorí sa tak, že v konzole, v mieste pracovného priečinku, sa napíše príkaz npm init. Keď sa neskôr bude inštalovať stiahnutý modul, jeho verzia aj názov budú vložené do tohto súboru. Meno tohto súboru je package.json a ako názov napovedá jeho formát je typu JSON. Tento formát bude objasnený neskôr. Package.json má hlavné využitie v prípade kedy je potrebné preniesť projekt na iný počítač respektíve server. S ním nie je nutné všetky moduly inštalovať zvlášť. Jediné čo je potrebné, je preniesť tento súbor, nainštalovať npm a potom v konzole v mieste pracovného priečinku spustiť príkaz npm install. Program npm potom nainštaluje všetky moduly do pracovného priečinku s verziami zaznačenými v package.json. Názov priečinku v ktorom sú všetky moduly umiestnené je node_modules.

1.5 MongoDB

MongoDB je bezplatný open-source databázový program s ľahkou škálovateľnosťou a jednoduchým extraktom dát.[7] Skupina záznamov sa volá kolekcia. Jediný záznam kolekcie sa volá dokument a položky v dokumente sa volajú polia. Jedná sa o NoSQL databázu, čo znamená, že jednotlivé dokumenty nemusia obsahovať rovnaké polia, čiže sa môže zmeniť jeho dátová štruktúra.

V NoSQL databáze je identifikačné číslo dokumentu na rozdiel od SQL databázy (ktorá začína od jednotky) dvanásť bitové číslo, z ktorého prvé štyri bity sú časová známka (time stamp), čo je počet sekúnd od 1. januára 1970. Ďalšie tri bity sú identifikačné číslo zariadenia (machine identifier), takže ak v rovnaký čas vytvoria záznam dva počítače každý bude mať iné identifikačné číslo. Ďalšie dva bity sú procesové bity (process id) a tak isto zabezpečujú vytvorenie jedinečného identifikačného čísla. Posledné tri bity identifikačného čísla sú náhodne vygenerované.[5] V tomto je výhoda vysokej škálovateľnosti a rýchlosti. Pri potrebe uložiť veľa dokumentov v jeden čas sa nemusí zisťovať, aké bolo identifikačné číslo posledného uloženého záznamu. Jednoducho sa vygeneruje nové jedinečné identifikačné číslo (pole označené ako _id) bez zbytočného zdržiavania. Navyše sa z identifikačného čísla dá určiť, kedy bol dokument vytvorený.

1.5.1 Ukladanie a čítanie z databázy

Pri práci s databázou sa neodporúča pridávať, odoberať, alebo upravovať údaje priamo v databáze cez nejaký prehliadací program. Keďže sú to surové dáta, ktoré majú mať presnú formu, je ľahké spraviť chybu a znehodnotiť dokument. Pre čítanie a zápis dokumentov je potrebné s databázou nadviazať spojenie. Databáza môže bežať lokálne, alebo sa k nej pristupuje cez internet pomocou autentifikácie. Pri spojení s databázou je potrebný port a IP adresa, ktorá je v prípade lokálnej databázy localhost. Pri pristupovaní cez internet je okrem IP adresy a portu potrebný aj spôsob prístupu, SSL, SSH alebo iné, od ktorého sa následne odvíja rozsiahlejšie nastavenie.

Pre obsluhovanie databázy existuje veľa knižníc a modulov na vyššie spomínanom npm servery. Jeden z najpoužívanejších je Mongoose. Jemu sa na začiatku programu zadefinuje schéma dokumentu, podľa ktorého má záznamy vytvárať a do ktorej databázy sa pripojiť. Keď je potom potrebné uložiť dokument do databázy, vytvorí sa nová inštancia z návratovej hodnoty funkcie, ktorá vytvárala schému. Naplní sa údajmi a zavolá sa nad ňou metóda save. MongoDB potom uloží dokument do určenej kolekcie. Ak kolekcia ešte

neexistuje, MongoDB ju automaticky vytvorí. Ak by bolo neskôr potrebné zmeniť polia dokumentu stačí zmeniť schému, pričom je možné stále ukladať do tej istej kolekcie.

Čítať, modifikovať, alebo mazať databázu je možné s veľa rôznymi funkciami. Tieto funkcie majú minimálne dva parametre. Prvý parameter je filter, ktorý selektuje dokumenty v kolekcii. Je to logická funkcia, ktorá používa syntakticky vlastné logické operátory. Druhý parameter je selekcia polí, ktoré chceme vrátiť s dokumentom. Ostatné parametre závisia od typu funkcie. Príklad asynchrónneho hľadania v databáze bude aj s ukážkou kódu vysvetlený neskôr. To, že NodeJS pracuje s databázou asynchrónnym spôsobom, má veľkú výhodu oproti prostrediam so synchrónnym spracovaním. V prípade, že je databáza na druhom konci sveta, server nemusí čakať na odpoveď, aby mohol spracovávať ďalšie požiadavky.

1.5.2 JSON

MongoDB databáza ukladá dáta vo formáte JSON z anglického výrazu JavaScript Object Notation.[6] Je to spôsob zápisu dát nezávislý na počítačovej platforme, určený primárne na prenos dát. Môže to byť ľubovoľná dátová štruktúra (číslo, reťazec, boolean, objekt alebo napríklad pole objektov) uložená ako reťazec UTF-8 znakov. Json používa príponu súboru .json a je často používaný kvôli jeho univerzálnosti a prenositeľnosti. Každý JSON začína a končí množinovými zátvorkami. Väčšinou sa na JSON prevádza objekt a z toho je odvodený aj jeho názov - JavaScript Object Notation. Každú dátovú štruktúru (podľa pravidiel prevodu) je možné previesť do formátu JSON, napríklad n-rozmerné pole. Dátová štruktúra, ktorá bola prevedená na JSON, môže byť prevedená späť. JavaScript má na to zabudované funkcie JSON.parse a JSON.stringify. Hlavné využitie prevádzania štruktúry JSON na objekt a späť je pri posielaní dát cez internet, alebo pri ukladaní objektu do databázy, kedy je možné dáta poslať iba ako reťazec UTF-8 znakov.

```
//ukážka JSON objektu uloženého v databáze
{
_id“ : ObjectId(“5be4de195a0a8211f8c5b4ac“),
“znacka“ : “škoda“,
“typ“ : “felicia“,
“rok“ : “2005“
}
```

1.6 Git

Git je systém pre správu verzií, ktorý ukladá históriu zmien a pomocou ktorého je možné nahrať pracovnú kópiu do druhého Git priečinku. S Gitom je možné docieliť ešte viac, ale tieto dve hlavné veci budú využité pri písaní, testovaní a distribúcií kódu. Originál Git nedisponuje grafickým rozhraním, ale otvára sa v príkazovom riadku. Po nainštalovaní Gitu je potrebné vytvoriť Git repozitár s príkazom git init, v priečinku ktorý má Git zálohovať a nastaviť vzdialené repozitáre (tzv. remote) príkazom git add remote. Vzdialené repozitáre sú taktiež priečinky, v ktorých bol inicializovaný Git. Do nich sa verzie kódu nahrávajú, alebo sa z nich sťahujú. Môžu to byť priečinky aj v tom istom operačnom systéme, alebo priečinky dostupné cez internet. Ak je remote dostupný z internetu, pristupuje sa k nemu jedným z dvoch protokolov, a to cez HTTPS protokol, alebo cez SSH protokol. Pri prístupe cez SSH je možné nastaviť SSH kľúč, s ktorým nahrávanie zmien do repozitára prebehne bez potreby zadať SSH meno a heslo. Jedným z remote je server GitHub, ktorý slúži pre zálohu verzií kódu. Ak by sa v budúcom kóde vyskytol problém, alebo by sa nejakou chybou prišlo o napísaný kód, stačí si stiahnuť staršiu verziu z tohto servera. Preto je vhodné si kód zálohovať tak často, aby medzi jednotlivými verziami nebol príliš veľký rozdiel. Pri zálohovaní kódu sa pracuje s príkazmi git commit, pre uloženie zmien do Git repozitára a git push s názvom uloženého remote repozitára, pre odoslanie zmien. Samozrejme, že je potrebné nastaviť aj druhý repozitár. V prípade GitHub repozitára to server urobí sám, ale pre repozitáre do ktorých sa budú posielať zmeny to je potrebné urobiť manuálne a to tak, že po inicializovaní Git repozitára sa zmení konštanta receive.denyCurrentBranch na updateInstead. S touto zmenou sa bude aktualizovať pracovná kópia v repozitári.

Nie ale všetky zmeny je dobré nahrávať na server. Buď sú zbytočné, ako napríklad pomocné súbory, ktoré si server pri práci vytvorí aj tak sám a tie pôvodné by prepísal, alebo sú to konštanty, ktoré musia ostať utajené. V týchto prípadoch sa vkladajú mená spomínaných súborov do súboru .gitignore. Tento súbor sa vytvoril po inicializácií repozitára a otvoriť sa dá ľubovoľným textovým editorom. Mená súborov, ktoré nebude Git zálohovať sa v ňom napíšu každé na nový riadok. Tento krok je nutný, pretože záloha na GitHub servery je verejná.

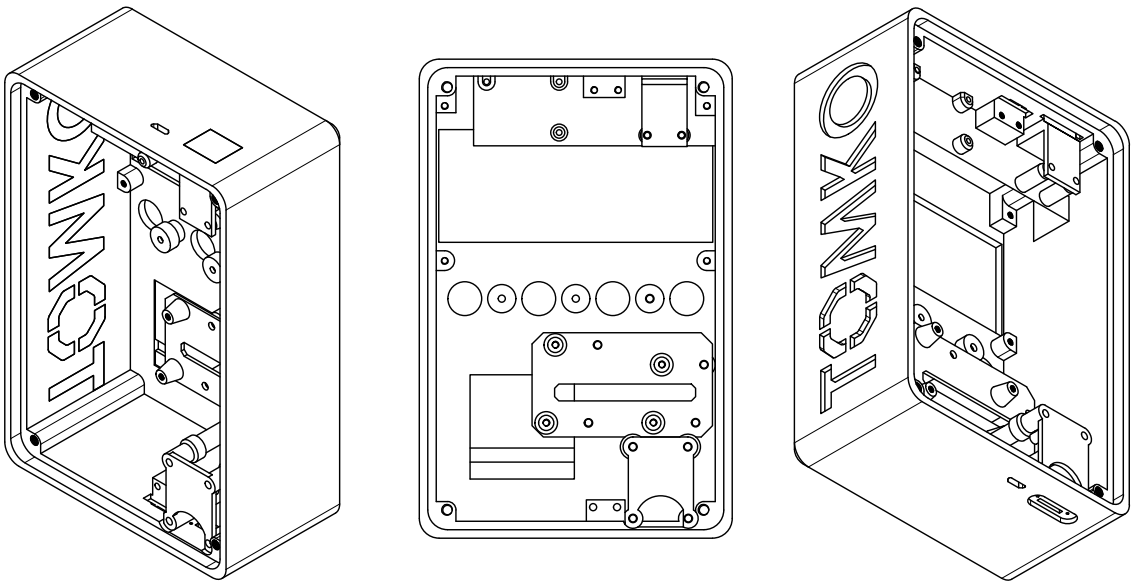
1.7 JSON web token

Dôležitý prvok v zabezpečení servera, na ktorý sa prihlasujú užívatelia, je JSON web token. Uložený je v cookies, ktoré sa odosielajú s každým requestom užívateľa. Takto server vie, že ide o overeného užívateľa, ktorý má oprávnenie na vykonávanie zmien. JSON web token obsahuje tri reťazce, na prvý pohľad nezmyselných znakov, oddelených bodkami. Jednotlivé reťazce sú hlavička, obsah a podpis tokenu. Hlavička je enkódovaný JSON s algoritmom base64url, v ktorej sú zvyčajne iba typ algoritmu (najčastejšie HS256) a typ tokenu (JWT ako skratka JSON web token). Obsah tokenu je tak isto enkódovaný JSON algoritmom base64url a obsahuje dáta, ktoré token prenáša. Môže to byť prihlasovacie meno, email, adresa, alebo niečo iné, čo oddeľuje jednotlivých užívateľov. Posledný reťazec je podpis. Je to všetko (enkódovaná hlavička tokenu s obsahom tokenu aj s bodkou medzi nimi) pred druhou bodkou enkódované algoritmom HMACSHA256 s použitím tajného kľúča, ktorý ovplyvní, ako podpis vyzerá. Takto je posledným reťazcom token zabezpečený. Keď sa používateľ prihlási, dostane vlastný token, s ktorým môže vykonávať len tie zmeny, na ktoré má oprávnenie. Užívateľ môže dekryptovať hlavičku tokenu aj jeho obsah, ale nemôže zmeniť podpis, ktorý overuje, že hlavička a obsah nebol zmenený, pretože nepozná tajný kľúč, s ktorým bol podpis vytvorený. Ak by si užívateľ zmenil obsah tokenu, nedokáže si ho podpísať a jeho request neprejde cez zabezpečenie servera.

2 Fyzická časť

2.1 Šasi

Šasi je navrhnutá v programe Fusion 360. Z vrchu sú otvory na štyri tlačidlá a jeden štvorriadkový LCD displej. Zozadu aj spredu je mikro-usb na napájanie. Spredu je otvor na SD kartu. Celkové rozmery šasi sú 170 na 111 na 60 milimetrov. Jej šírka je určená šírkou displeja a dĺžka a výška sú určené tak, aby bol dostatočne veľký pracovný prístup k jednotlivým modulom. Ukážku šasi je možné nájsť na obrázku 2.1-1. Použitý materiál na 3D tlač bol polylactic acid (PLA), s teplotou tavenia 180 °C a teplotou skleného prechodu 65 °C. Teplota trysky bola nastavená na 215 °C. Z vonkajšej strany bola nastriekaná tmelom, prebrúsená a nafarbená na bielo. Zospodu je plexisklový kryt, vyrezaný laserom, na ktorom sú napísané niektoré základné informácie, ako fyzická adresa zariadenia, rok výroby a podobne. Kryt sa prikrúca štyrmi skrutkami v každom rohu. Na prikrútenie modulov boli použité skrutky s metrickým závitom o veľkosti 2,5 mm. Možnosti ako vyrobiť závit v plaste je niekoľko. Buď sa vymodeluje a vytlačí o niečo menšia diera, do ktorej sa zakrúti skrutka nasilu, alebo sa závit vymodeluje a vytlačí, čo s presnosťou bežnej 3D tlačiarne od metrického závitu 2 mm a vyššie nie je problém, alebo sa závit do plastu vyreže závitorezom. Ďalšia možnosť je, že sa kúpi a do diery zataví mosadzný závit. Očakáva sa, že šasi nebude mechanicky namáhaná. Preto sú závity riešené druhou možnosťou, teda vymodelovaným závitom.



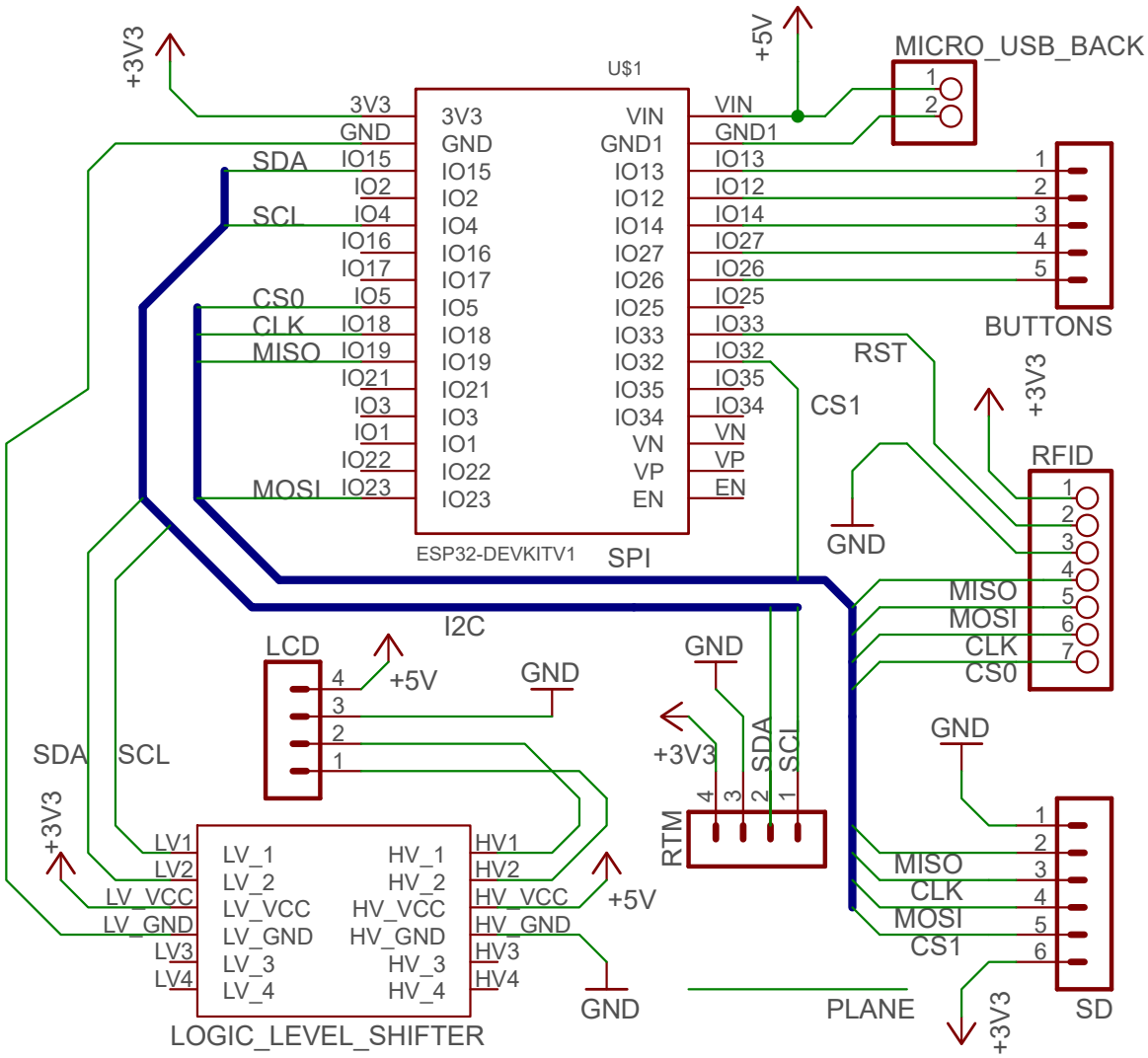
Obr. 2.1-1 Šasi

2.2 Napájanie

Zariadenie neobsahuje zdroj, ale je napájané piatimi voltami cez mikro-USB konektory. Jeden konektor je spredu a druhý zozadu. Konektor spredu je spojený s USB zbernicou radiča, ktorý komunikuje s mikrokontrolérom cez UART, takže cez predný konektor sa dá mikrokontrolér aj preprogramovať. Predvolene by sa malo zariadenie napájať zo zadného konektoru, pretože napájanie z neho je priamo pripojené na VIN pin dosky mikrokontroléru, z ktorej sa rozvádza 5 V napájanie po zariadení a stabilizuje sa na 3,3 V pre mikrokontrolér. Predný konektor sa pripája na VIN cez ochrannú diódu, ktorá spôsobuje úbytok napätia, kvôli čomu sa LCD displej pri väčšom odbere mikrokontroléru striedavo stmavuje. Táto dióda je súčasťou dosky, na ktorej je mikrokontrolér prispájkovaný. Je tam z dôvodu ochrany 5 V napätia pripojeného USB zariadenia k radiču, pretože na VIN pin sa môže priviesť napätie až do 12 V. Mikrokontrolér je napájaný zo stabilizátora napätia ASM1117-3.3. Tento čip[15] s prúdovou výdatnosťou 1,5 A znižuje napätie na 3,3 V, ktorým napája aj ostatné 3,3 V moduly. Jeho prúdový odber v hlavnej funkcií je priemerne 180 mA pri 5 V napätí, čo znamená že jeho spotreba je priemerne 0,9 W. Zariadenie je navrhnuté s tým, že nebude pripojené na batériu ale do napájacej siete.

2.3 Moduly

Ako už bolo spomenuté v úvode a v teoretickej časti, riešenie tohto dochádzkového systému je postavené na moduloch, ktoré majú výhodu nie len v jednoduchej oprave a cene, ale aj v rýchlom riešení hardvérovej časti. Dochádzkový systém je komplexná vec a ak by mal byť konštruovaný od začiatku a z tých najmenších stavebných blokov a súčiastok, s určitosťou by ho nebolo možné stihnúť dokončiť. Celá schéma zapojenia je na obrázku 2.3-1. Doska s mikrokontrolérom nie je priamo prispájkovaná k rozvodnej doske, ale je s ňou spojená dutinkovými konektormi. Tak isto žiadny z modulov nie je prispájkovaný priamo k doske, ale všetky sú s ňou spojené pomocou prispájkovaných vodičov. Takto je ťažšie zariadenie rozobrať, ale zato je spojenie dosky s modulmi spoľahlivejšie, ako cez konektory. Jedine prevodník napätia logických úrovní je k doske prispájkovaný. Je to doska plošného spoja, s tranzistormi a pasívnymi súčiastkami, prispájkovaná ako SMD súčiastka. Vodič PLANE pokrýva nevyužité miesto na plošnom spoji, ktoré tým pádom nie je nutné odleptať.



Obr. 2.3-1 schéma zapojenia

2.3.1 Čítačka kariet

Čítačka je umiestnená na hornej strane šasi, 2 milimetre pod vrchnou časťou. Obslužný čip na tomto module je MFRC522. On sa stará o všetku komunikáciu s kartou a mikrokontrolérom.[16] Frekvencia elektromagnetického poľa, po ktorej komunikuje s RFID čipmi, je 13,56 MHz. S mikrokontrolérom komunikuje cez SPI zbernicu. Na napájanie potrebuje 3,3 V napätie, ktoré má z dosky mikrokontroléru. Čip je schopný čítať kary a tagy s čipom z rodiny MIFARE Classic od spoločnosti NXP. Karta, alebo tag, môže uschovať iba 1 KiB dát, navyše prečítať celý obsah trvá približne 3 sekundy, čo je príliš dlhá doba pre akúkoľvek využiteľnosť úložného priestoru týchto čipov v tomto dochádzkovom systéme. Pre dochádzkový systém, však stačí len identifikačné číslo, ktoré sa dá prečítať do 100 milisekúnd.

2.3.2 Displej

Displej je štvorriadkový typu LCD. Má na sebe radič PCF8574T, ktorý komunikuje s mikrokontrolérom cez I^2C zbernicu a zapisuje do pamäte displeja. Takto nie je potrebné viesť šestnásť vodičov k displeju, ale stačia len štyri. Jeho I^2C adresa je 0x27. Podsvietený je na modro a jeho kontrast bol nastavený potenciometrom na zadnej strane jeho dosky. Displej potrebuje na napájanie 5 V a logická úroveň komunikácie je tiež 5 V, takže medzi displejom a mikrokontrolérom je prevodník napätia logických úrovní.

2.3.3 SD modul

SD modul je doska plošného spoja s puzdrom na SD kartu a pasívnymi súčiastkami. SD karta potrebuje napätie od 2,7 V do 3,6 V, takže je napájaná priamo z dosky mikrokontroléra. Logická úroveň komunikácie je v rovnakom rozsahu, takže nie je potrebné použiť prevodník logických úrovní. S kartou sa komunikuje cez SPI zbernicu a slúži na ukladanie kópií záznamov, ktoré sa odosielajú na server. V prípade výpadku internetu je možné záznamy prehliadať priamo z nej. Avšak záznamy vytvorené cez počítač sa na kartu nezálohujú.

2.3.4 Mikro-usb modul

Mikro-USB modul je doska plošného spoja s prispájkovaným mikro-USB konektorom a otvormi na priskrutkovanie k šasi. Vo vnútri šasi sú dva tieto moduly. Jeden je zozadu na napájanie a druhý je spredu, na aktualizáciu programu mikrokontroléru.

2.3.5 RTC modul

Modul skutočného času (z anglického slova Real Time Clock) v sebe udržiava aktuálny čas, aj pri odpojenom napájaní zariadenia.[10] Používa na to ploskú gombíkovú batériu, s napätím 3 V. Čas inkrementuje čip DS3231, ktorý má v sebe teplotne kompenzovaný kryštálový oscilátor, kvôli čomu dosahuje vysokú presnosť a to ±2 ppm pri teplote 0 až 40 °C. To znamená, že za rok sa môže odchyľovať o ±60 sekúnd od skutočného času. V programe sa ale aktualizuje každý deň, takže takto veľkú odchýlku nikdy nedosiahne. Počítať čas môže do roku 2100. Má aj funkcie, ktoré nebudú využité a to programovateľný obdĺžnikový signál a čítanie teploty. Na doske modulu je EEPROM pamäť, ktorú používa DS3231 na zapisovanie času. Zariadenie potrebuje na napájanie napätie od 2,3 V do 5,5 V. Na napájanie preto stačí 3,3 V, aby medzi RTC a mikrokontrolérom nemusel byť prevodník logických úrovní. RTC komunikuje cez I^2C zbernicu a jeho adresa je 0x68.

2.4 Inicializácia

Zariadenie nemá tlačidlo na zapnutie ale zapína sa ihneď po pripojení mikro-USB kábla. Po zapnutí sa ako prvé nastavia vstupné piny na ktorých sú pripojené tlačidlá. Keďže tieto registre nemajú nejaký vplyv na nastavenie ostatných prvkov v programe, môžu sa nastavovať kdekoľvek v inicializačnom kóde. V tomto prípade je to hneď na začiatku. Ako druhé sa inicializujú registre, pre sériovú komunikáciu cez UART s počítačom. Táto zbernica je iba pre dva zariadenia, ktoré medzi sebou komunikujú, takže jej nastavenie sa už v programe nebude meniť. Jej inicializácia taktiež neovplyvní budúce chovanie programu, čiže by mohla byť inicializovaná tiež v ľubovoľnom mieste, ak by nebola potrebná na odosielanie dát o ďalšom stave programu. Preto musí byť inicializovaná skôr, ako ostatné moduly, s ktorými by mohla inicializácia teoreticky zlyhať. Napríklad kvôli absencií SD karty. Zbernica je nastavená na rýchlosť prenosu, ktorou mikrokontrolér predvolene odošle informáciu o chybe a výpis z registrov pri zlyhaní jeho programu. Je to 115200 bitov za sekundu. Častá chyba, pri ktorej program zlyhá, je pre pretečenie premenného poľa, alebo pri použití knižníc, ktoré pracujú s tými istými perifériami. Netreba sa ale nechať pomýliť. Mikrokontrolér odošle informácie pri zlyhaní programu vždy a to aj keď je zbernica nastavená na inú prenosovú rýchlosť, alebo nie je vôbec inicializovaná. Dôvod, že je nastavená na rovnakú rýchlosť akou odosiela informácie o chybe, je kvôli počítaču, ktorý má tiež pevne nastavenú rýchlosť čítania z prenosu. Ak by ju mal inú ako 115200, po zlyhaní programu by sa v konzole za posledným textom namiesto výpisu chyby objavili otázniky a nezmyselné alebo neviditeľné znaky.

Po inicializácií komunikácie s počítačom nasleduje inicializácia LCD displeja. Tá musí ísť ako druhá. Keďže dochádzkový systém nebude vždy pri zapínaní pripojený k počítaču, nebude môcť užívateľ vedieť o prebiehajúcom stave inicializácie inak, ako cez displej. Ako tretia v poradí nasleduje inicializácia SD karty. Pri jej inicializácií je riziko najväčšej neúspešnosti. Karta môže mať zlý kontakt, byť povysunutá, alebo chýbať úplne. SD karta komunikuje cez SPI zbernicu, na ktorej je ešte čítačka kariet, konkrétne s čipom rc255. Dôležité je kartu inicializovať skôr, ako iné zariadenie na zbernici. SD karta dokáže komunikovať v dvoch módoch. V móde SD a v móde SPI. Predvolene komunikuje v móde SD [12]. Ak by sa najskôr inicializoval rc522, karta by rušila komunikáciu a inicializácia by neprebehla. Pre to je potrebné najskôr inicializovať SD kartu, aby sa prepla do módu SPI. Potom bude reagovať na signál zo zbernice len v tom prípade, keď mikrokontrolér stiahne jej SS na zem (podmienka komunikácie cez SPI). Ako štvrtý je teraz možné

inicializovať čip rc522, ktorý obsluhuje anténu a číta priložené karty a tagy. A ako posledný modul sa inicializuje RTC, ktorý v sebe počíta čas. Po jeho inicializácií sa ešte prečíta napätie na jeho batérií. Ak bude príliš nízke na displeji sa zobrazí upozornenie. Po inicializácií všetkých modulov sa mikrokontrolér pripojí na WiFi. Toto je druhý bod v inicializačnej časti programu so zvýšeným rizikom zlyhania. V ďalšom programe sa nedá pokračovať, pokiaľ pripojenie neprebehne úspešne. Aj keby bolo možné zálohovať záznamy na SD kartu počas absencie WiFi siete a po opätovnom pripojení ich poslať na server, počíta sa s tým, že prístup k internetu bude permanentný a chyba v pripojení sa vyrieši ihneď na mieste.

S prístupom na internet sa ako prvý odošle request, z ktorého odpovede sa aktualizuje čas v zariadení. Ak je server nedostupný, alebo spojenie z neznámeho dôvodu zlyhá, zariadenie stiahne čas z modulu skutočného času. Ak by čas stiahnutý z RTM bol neaktuálny, program to zaznamená v hlavnej funkcií, hneď po inicializácii. Počas celého inicializačného kódu sa progres inicializácií zobrazuje na displeji. Na konci sa na displeji zobrazí domovská obrazovka.

2.5 Ovládanie a navigácia

Predvolene sa na displeji zobrazuje čas, dátum, druh dochádzky, ktorá sa odošle po priložení kartičky na server a jej poznámka. Druh dochádzky môže byť len príchod, alebo odchod. V prípade odchodu si môže pracovník vybrať jednu z predvolených poznámok, ako napríklad lekár, alebo obed. Pri každom stlačení tlačidla sa ozve krátke kliknutie piezoelektrického kryštálu ako signalizácia zaznamenaného stlačenia. Tlačidlá majú odozvu 200 milisekúnd kvôli zákmitom. V prípade, že sa objaví nejaká chyba na zariadení, ozve sa sekundové pípnutie.

Dochádzkový systém sa ovláda pomocou štyroch tlačidiel z hornej strany. Prvé dva sú na navigáciu hore a dole, tretie je zrušiť a štvrté je potvrdiť. Na úvodnej obrazovke sa na voľbu medzi druhom dochádzky a jej poznámkou používajú tlačidlá na navigáciu hore a dole. Toto je jediná vec ktorú je možné zmeniť na úvodnej obrazovke. Pridržaním tlačidla potvrdiť sa zobrazí menu. V menu sú na výber možnosti zobraziť číslo kartičky po priložení a zobraziť informácie o stave hardvéru, kde je MAC adresa, stav o kapacite baterky pre zálohovanie času a voľné miesto na SD karte. Po určitom čase používania dochádzkového systému budú niektoré prvky z navigácie zmenené.

2.6 Hlavná funkcia

Hlavná funkcia tzv. main sa opakuje v slučke. Obsahuje podfunkcie, ktoré budú vykonané na základe splnenia jednotlivých podmienok. Na jej konci je funkcia delay s argumentom 1, ktorá zaručí, že sa hlavná funkcia nebude opakovať skôr, ako po jednej milisekunde. Ešte pred začiatkom hlavnej funkcie je inicializovaná premenná s názvom counter. Prvá podfunkcia ho porovnáva s hodnotou 15000, čo je približne 15 sekúnd. Ak je hodnota v premennej counter menšia, tak sa counter inkrementuje, ak nie je, tak sa vynuluje. V prípade keď sa nuluje counter, ešte sa zobrazí na displeji pôvodné nastavenie. To nesúvisí s časovaním a bude to vysvetlené na konci tejto podkapitoly. Takto je možné s matematickou funkciou modulo približne časovať ostatné podfunkcie, keďže counter s časom lineárne rastie a vynuluje sa približne každých 15 sekúnd. Tu je príklad kódu, ktorý odošle textové pole na sériový port približne každých 30 milisekúnd.

```
if ((counter % 30) == 0) {
Serial.println(“Hello World“);
}
```

Samozrejme toto časovanie nie je dokonalé a pre presné časovanie je vhodné použiť vnútorný časovač. Navyše s ním nie je možné časovať funkciu s rozostupom napríklad 10 sekúnd, pretože po 15 sekundách sa counter vynuluje a znova bude počítať ďalších desať sekúnd. Až potom by sa funkcia vykonala, čo by vo výsledku dalo 15 sekúnd a nie 10. Pre malé časy a prípady kedy presné časovanie nie je nutné, to však postačuje.

Ostatné podfunkcie majú na starosť nasledovné. Každých 800 milisekúnd sa stiahne čas z RTM modulu a vypíše sa na displej. Je dôležité by bol časový interval výpisu času na displej menší ako 1 sekunda aby sa nestalo, že by sa čas na displeji inkrementoval o dve sekundy namiesto jednej. Každých 125 milisekúnd sa skontroluje, či čítačka nezaznamenala kartu. Každé 3 sekundy sa skontroluje, či má mikrokontrolér spojenie cez WiFi. Ak ho stratil, pokúsi sa znova pripojiť. Ak sa to nepodarí, vypíše chybu na displej. Na displeji sa zobrazí možnosť znova sa pokúsiť o pripojenie. Zvyšné tri pod-funkcie nie sú časované. Prvá overuje aktuálny čas, ktorý sa sťahuje z RTM. Ak čas nie je aktuálny, napríklad kvôli vypnutiu dochádzkového systému v čase keď v RTM nebola baterka, pokúsi sa dochádzkový systém znova aktualizovať čas zo serveru. Ak sa to nepodarí, vypíše chybu na displej a ďalej nie je možné robiť záznamy. Na displeji sa zobrazí možnosť znova sa pokúsiť o aktualizovanie času. Ostatné podfunkcie testujú stlačenie tlačidiel. Prvým tlačidlom je možné meniť medzi príchodom a odchodom. Druhým je možné vybrať

poznámku k odchodu (obed, lekár a podobne). A štvrtým tlačidlom je možné vojsť do menu, kde sú funkcie pre zobrazenie identifikačného čísla kartičky, voľného miesta na SD karte a napätia baterky modulu skutočného času.

Predpokladá sa, že väčšina zamestnancov má svoj príchod medzi polnocou a dvanástou hodinou a odchod medzi dvanástou hodinou a polnocou. Preto dochádzkový systém sám mení druh dochádzky v prvej časti dňa na príchod a v druhej časti dňa na odchod. Môže sa ale stať, že bude potrebné urobiť záznam, ktorý druhom neodpovedá jeho času. Preto je možné meniť druh dochádzky. Avšak zamestnanci počítajú s tým, že ráno je nastavený príchod automaticky, rovnako ako odchod po obede. Preto ak niekto zmení druh dochádzky, premenná counter sa vynuluje a po prekročení 15 sekúnd pri nulovaní premennej sa druh vráti na predvolenú hodnotu vyplývajúcu z času. Druh sa vynuluje aj po priložení kartičky.

2.6.1 Formát dochádzky a jej odosielanie

Po priložení karty sa prečíta jej identifikačné číslo. Zariadenie sa potom pokúsi nadviazať klientské spojenie so serverom. Pri neúspechu kontaktovať server, zariadenie túto informáciu oznámi na displeji. Ak sa spojenie podarilo nadviazať, mikrokontrolér zostaví JSON, v ktorom je číslo karty, druh dochádzky, v prípade odchodu aj jeho bližšia špecifikácia, pôvod ktorý určuje, že záznam pochádza z mikrokontroléra a pole s názvom force. To určuje, či pri opakovanom druhu záznamu má byť napriek tomu tento záznam vložený. Potom sa JSON odošle na server. Ak server odpovie, že sa záznam podarilo uložiť, mikrokontrolér ho uloží aj na SD kartu ako zálohu. Nie však vždy, po úspešnom nadviazaní spojenia so serverom, všetko ostatné prebehne bez problémov. V kóde je potrebné ošetriť najpravdepodobnejšie zlyhania, ktoré by mohli nastať. Najväčšie riziko je priloženie neznámej RFID karty. Server vtedy odpovie, že kartu nepozná a mikrokontrolér to vypíše na LCD. Ďalšia chyba je spôsobená vložením rovnakého druhu dochádzky do databázy. Vtedy sa mikrokontrolér spýta či chce užívateľ naozaj vložiť opakovaný záznam. Ak sa rozhodne pre áno, odošle sa JSON so spomínaným force poľom. Toto môže zamestnanec využiť napríklad vtedy, ak vie, že ráno si nechtiac zaznamenal odchod. Ostatné chyby sú prerušenie spojenia, neuloženie dochádzky na SD kartu, neschopnosť pripojiť sa k serveru a neznáma chyba ktorá pokrýva ostatok. Ak zlyhá niečo iné okrem mikrokontroléru známych chýb, vypíše sa, že nastala neznáma chyba. Nie je totiž možné ošetriť úplne všetky možnosti do detailu. Avšak tie ktoré sú ošetrené, tvoria drvivú väčšinu prípadov, ktoré môžu nastať.

3 Serverová časť

3.1 Zabezpečenie

3.1.1 Pripojenie

Každý request, ktorý príde na server, prejde autentifikáciou. Ten pošle request ďalej, ak v cookies nájde platný JSON web token, v skratke iba token. Token dostane každý, kto sa prihlási do dochádzkového systému platnými údajmi. Jeho platnosť je na 9 hodín. Po deviatich hodinách neaktivity je token neplatný a užívateľ sa musí znova prihlásiť. Ak server našiel v requeste token, k jeho odpovedi pridá nový token s novou platnosťou na 9 hodín dopredu a prepíše ten starý. Takto sa pri aktivite používateľa na servery nestane to, že ho server odhlási. Token obsahuje meno, priezvisko, email a rolu. Rola určuje, či má daný užívateľ oprávnenie administrátora. Ak ho má, môže vykonávať zmeny a spravovať dochádzku zamestnancov. Ak ho nemá, nedokáže sa prihlásiť na server. V budúcnosti je možné pridať zobrazenie vlastnej dochádzky pre neadministrátorských užívateľov. Token je zabezpečený tajným kľúčom. Tento kľúč je spolu s ostatnými konštantami v súbore process.env ktorý nie je zálohovaný na serveri GitHub.

Nie všetka komunikácia so serverom si vyžaduje token umiestnený v cookies. Výnimkou je vytváranie hesla z odkazu, ktorý prišiel užívateľovi na email. Token je vtedy v URL adrese. Tak sa overí, že heslo vytvára oprávnený užívateľ. Ďalšou výnimkou je mikrokontrolér, ktorý nedisponuje tokenom, pretože sa neprihlasuje cez stránku. On pristupuje na server cez jeho vlastný kľúč. Samozrejme ak by niekto zistil tento kľuč, mohol by odoslať dochádzku, ktorú by následne server uložil. Preto je tento kľúč súčasťou súboru process.env a do mikrokontroléru je vpísaný manuálne. Navyše s týmto kľúčom sa dá komunikovať iba po cestách, ktoré využíva mikrokontrolér. Preto s ním nie je možné napríklad vymazať užívateľa.

Ak request neprejde cez overenie a je typu GET, server mu odošle prihlasovaciu stránku. Ak je to iný request ako GET, server mu odošle iba text o neoprávnenom prístupe a zmení HTTP status kód na 401, čo znamená neopravený prístup. Pri prihlasovaní je nutné vyplniť email a heslo. Prihlasovacie údaje sa potom odošlú cez POST HTTPS request. Heslo sa odosiela doslovne, ale po prijatí serverom je hneď zahešované algoritmom SHA256. Z bezpečnostných dôvodov je heslo v databáze uložené len v zahešovanej

podobe. Prijaté reťazce sa potom porovnajú s tými v databáze. Pri zhode sa odošle token a úvodná stránka, pri nezhode sa odošle oznam o nesprávnych prihlasovacích údajoch.

Okrem vytvorených účtov je možné sa prihlásiť na server aj administrátorským účtom. Tento účet je potrebný pre prvé prihlásenie do systému. Jeho prihlasovanie má na starosti iná časť kódu ako pre vytvorené účty. Server po prvom prihlásení vytvorí databázový záznam administrátorského účtu kvôli tomu, aby sa aj na administrátorskom účte dali ukladať nastavenia ako profilový obrázok a filter dochádzky. Tento účet sa nezobrazuje nikde v dochádzkovom systéme. Ak by aj niekto tento účet z databázy manuálne vymazal, pri ďalšom prihlásení sa vytvorí nový s predvolenými nastaveniami. Prihlasovacie údaje do tohto účtu sú v súbore process.env ktorý nie je zálohovaný na serveri GitHub.

Server načúva na oboch portoch pre HTTP (80) aj HTTPS (443). Komunikácia však prebieha iba cez HTTPS pripojenie. Pre pripojenie sa cez HTTPS server potrebuje disponovať privátnym kľúčom, z ktorého sever vygeneruje verejný a ten posiela každému, kto sa chce pripojiť. Privátny kľúč nie je zálohovaný na GitHub serveri a nesmie byť nikomu dostupný. Okrem kľúča je ešte potrebný certifikát. Certifikát sa odosiela spolu s verejným kľúčom. Tretia strana potom overí, že certifikát a verejný kľúč patria k sebe. Zabezpečenie spočíva v tom, že útočníci nemôžu mať rovnaký pár verejný kľúč s certifikátom, pretože ak by odoslali rovnaký verejný kľúč ktorý patrí k tomu certifikátu, nie sú schopný dešifrovať prichádzajúce správy, pretože nemajú privátny kľuč ktorý patrí k tomu verejnému. Ak teda zvolia vlastný kľúč, s ktorým vedia dekryptovať ním zakryptované správy, tretia strana im neoverí certifikát a prehliadač označí stránku za nebezpečnú. Na vygenerovanie privátneho kľúča a certifikátu bol použitý openssl program.

3.1.2 Serverové konštanty

Preto aby server vedel odosielať emaily, generovať tokeny, alebo vykonávať inú činnosť závislú od overovania, alebo inak súvisiacu so zabezpečením, potrebuje serverové konštanty. Môžu to byť napríklad prihlasovacie údaje alebo len náhodné reťazce znakov použité na šifrovanie a dešifrovanie citlivých informácií. Avšak musia byť už od začiatku programu k dispozícii. Tieto konštanty nemôžu byť súčasť kódu. Za prvé z bezpečnostných dôvodov, pretože kód je verejne dostupný, a za druhé kvôli univerzálnosti a prenositeľnosti kódu. Ak by totiž tieto konštanty boli natvrdo napísané v kóde a chcel by tento kód využiť niekto iný, musel by ich hľadať a prepisovať. To by bolo zbytočne náročné a zdĺhavé.

Takto sú na jednom mieste prehľadne uložené v pracovnom priečinku s názvom process.env. Názov tohto priečinku je vložený do priečinku .gitignore aby sa neverejné konštanty nedostali na GitHub server. Priečinok process.env obsahuje všetky konštanty okrem SSL privátneho kľúča a SSL certifikátu. Tieto konštanty sú každá ako samostatný súbor, pretože sú dlhé a nepotrebujú byť nahrané do globálnych konštánt, pretože sa v kóde používajú iba raz a to pri spúšťaní servera na HTTPS porte. Ostatné konštanty sa hneď po začatí programu nahrajú do globálnych premenných programu NodeJS. Konkrétne do procesného prostredia. Konštanty nahrá do prostredia na začiatku kódu modul dotenv, tento modul je ako každý modul stiahnutý z npm[14]. Súbor, z ktorého sa konštanty nahrajú do kódu, musí mať zakončenie .env a musí byť písaný v textovom režime bez medzier. Po nahratí konštánt sa k nim pristupuje podľa ich mena cez cestu process.env.meno_konštanty. Tu je príklad niektorých serverových konštánt.

```
HTTPS_PORT=3000
HTTP_PORT=2999
SECRET='p1oP}];{r<S^_p8qp<2wg^fM`M<(3
MCU_KEY=PzSesF6YJJVfrcu4cxlw0Pe9DQFM1Zba
```

3.2 Back-end

3.2.1 Server

Server je napísaný v JavaScripte a je uložený v súbore s názvom server.js, spúšťa sa spomínaným programom NodeJS. Po spustení sa ako prvé nahrajú do globálnych premenných konštanty zo súboru process.env. Po tomto sa nahrajú všetky moduly potrebné pre funkciu serveru, ich zoznam a funkcia bude prebratá neskôr. Nasleduje deklarovanie objektu, s ktorým sa odosielajú emaily. Toto je prvé miesto, kde sa použijú konštanty zo súboru process.env. Ďalej sa deklarujú schémy pre ukladanie účtov a dochádzky do databázy. Zo zadefinovanými schémami sa server pripojí cez modul Mongoose do databázy. Databáza beží lokálne, takže nie je nutné sa do nej prihlasovať. Nasleduje nastavovanie modulu express, tento modul prijíma, spracováva a odosiela requesty. Requesty však nespracováva len sám express, ale aj middleware. To je kód, ktorý sa vloží do expresu a tým sa rozšíri jeho funkcionalita. Najpotrebnejší middleware je parser, ten analyzuje request a vyrobí z neho objekt. Iný middleware zase overuje tokeny v requestoch. Middleware je možné stiahnuť z npm alebo, v prípade jednoduchého middleware kódu, si napísať vlastný. Za nastaveným express modulom sa zadeklarujú

funkcie na posielanie emailu a generovanie tokenu. Po tomto sa namapujú všetky cesty, na ktoré bude server odpovedať a následne sa spustí.

Ak sa užívateľ chce pripojiť na server, môže v prehliadači zadať, alebo kliknúť na adresu s prefixom https://, alebo bez neho. Prehliadač v prípade prefixu https:// automaticky doplní port, na ktorý sa request odošle na 443, čo je zaužívaný HTTPS port. Ak užívateľ nezadá prefix pre HTTPS pripojenie, prehliadač odošle HTTP request na port 80. V obidvoch prípadoch sa však očakáva, že sa užívateľ dostane na požadovaný server. Z tejto podmienky vyplýva, že server musí sledovať obidva porty. Keďže komunikácia so serverom dochádzkového systému musí prebiehať po zabezpečenom pripojení, po pripojení sa na port 80 cez HTTP server užívateľa presmeruje na HTTPS pripojenie, čo si užívateľ väčšinou ani neuvedomí.

Na to aby server odpovedal na jednotlivé cesty je ich potrebné v kóde namapovať. Táto časť je v súbore server.js najrozsiahlejšia. Cesty sa mapujú asynchrónne, čo je veľká výhoda NodeJS, ako bolo spomenuté už v teoretickom úvode.

```
//ukážka namapovanej cesty modulu express
app.get('/getaccounts', (req, res) => {
  accounts.find({
    _id: { $ne: ADMIN_ID }},{
    password: false,
    settings: false,
  }).then((doc) => {
    res.setHeader('x-status', 'ok');
    res.send(doc);
  }, (e) => {
    res.setHeader('x-status', 'cannot browse accounts database');
    res.send();
  });
});
```

Adresa cesty /getaccounts v príklade znamená, že ak niekto zadá do prehliadača IP adresu (alebo doménové meno) s touto cestou, server začne prehľadávať kolekciu s uloženými účtami a vráti každý účet okrem toho, ktorý má _id admina. Ak vyhľadávanie prebehne úspešne, server odošle pole dokumentov vo formáte JSON a nastaví položku v hlavičke odpovede x-status na konštantu “ok“. Ak vyhľadávanie skončí neúspešne (napríklad kvôli prerušenému spojeniu s databázou), odošle sa hlavička s položkou x-status s hlásením o nedostupnosti databázy.

Ako už bolo spomenuté, server, pre jeho plnú funkcionalitu pri svojej práci, potrebuje moduly, inak povedané knižnice. Dotenv na nahranie serverových konštánt z priečinku process.env do procesného prostredia. HTTP, HTTPS a express moduly na vytvorenie inštancie NodeJS servera pre zabezpečené, aj nezabezpečené pripojenie. Potrebuje tiež jsonwebtoken pre kryptovanie a dekryptovanie tokenov, z modulu crypto-js funkciu SHA256 na hešovanie hesiel, cookie-parser na analyzovanie prijatých cookies, body-parser na vytvorenie objektu z prijatého requestu, fs pre zapisovanie a čítanie lokálnych súborov, Mongoose pre prácu s databázou, serve-favicon na vlastnú ikonku v záložke prehliadača a nodemailer na posielanie emailov. V server.js sú nahraté aj iné moduly, ale momentálne nie sú využité. Celý kód serveru aj stránok je zálohovaný na stránke GitHub a verejne dostupný[11].

3.2.2 Databáza

Databáza beží na rovnakom hardvéri ako server. Jedná sa o databázový program MongoDB, ktorý je možné zadarmo stiahnuť z online stránky[13]. Knižnica, ktorá je použitá na prácu s databázou sa volá Mongoose. Je stiahnutá, ako aj ostatné knižnice, z npm. V kóde nie je nutné vytvárať novú kolekciu, MongoDB ju vytvorí, keď sa do nej zapíše prvý dokument. Server v databáze používa 2 kolekcie. V prvej sú uložené dokumenty jednotlivých účtov. Každý účet tu má vlastný záznam, teda dokument, v ktorom sú polia _id záznamu (používateľa), meno, priezvisko, email, zahešované heslo hešovacím algoritmom SHA256, číslo karty, rola a nastavenia účtu. Rola určuje či je daná osoba administrátor, alebo nie a v nastaveniach účtu sú konštanty filtra dochádzky. Funkcia ostatných polí vyplýva z ich názvu. Druhá kolekcia obsahuje dokumenty o dochádzke. Záznam o dochádzke obsahuje _id záznamu, _id používateľa ktorý záznam vytvoril, druh dochádzky (buď príchod alebo odchod), poznámka (lekár, obed a podobne), dátum a pôvod. Ten hovorí o tom, či bol záznam vytvorený priložením kartičky o čítačku, alebo vložením cez počítač.

3.3 Front-end

3.3.1 HTML a CSS

Pre vytvorenie obsahu stránky bol využitý HTML5 a CSS3. Názov záložky v prehliadači je attendace systems. Všetky identifikátory (id) aj class selektory sú v angličtine. Identifikátor elementu, ktorý mení svoj vnútorný obsah pre jednotlivé podstránky, má meno content. Stránka bola navrhnutá tak, aby bol jej obsah adaptívny aj pre mobilné zariadenia (tzv. responzívny dizajn). Elementy sa dokážu prispôsobiť od šírky 260 px. Pre responzivitu bolo využité CSS pravidlo @media screen. Takto nie je potrebné ošetrovať zmenu veľkosti elementov v JavaScripte, ale o zmenu rozlíšenia sa stará CSS kód ktorý to dokáže vykonávať rýchlejšie.

3.3.2 JavaScript

Stránka dochádzkového systému je riešená ako single page webside, čo znamená že hlavný HTML kód je len jeden a jeho podstránky mení až vlastný JavaScript, ktorý HTML podstránky vyžiadava a doplňuje sám namiesto prehliadača. Obyčajné stránky majú pre každú namapovanú cestu vlastný HTML kód. V tomto prípade hlavné HTML ostáva, pričom sa jeho časť doplní zo servera, poprípade sa len schová nežiadúca časť a zobrazí sa iná, už predom načítaná. Celé HTML sa preto obnoví až po kliknutí na refresh tlačidlo prehliadača. Toto je spôsob riešenia stránky, ktorý urýchli jej prehliadanie. Ak by mal každú stránku sťahovať server celú aj so všetkými zdrojmi, ktoré k tomu potrebuje, trvalo by mu to o stovky milisekúnd viac. Presný čas záleží na rozsahu stránky a rýchlosti pripojenia.

Každá podstránka má vlastný JavaScript kód, ktorý je potrebné nahrať zvlášť pri jej vyžiadaní. Hlavná stránka obsahuje hlavný JavaScript kód, ktorý vykonáva (mimo iných) aj túto funkciu. Ak sa stránka načítava cez prehliadač, ten automaticky vloží JavaScript do zdrojov a následne sa spustí. Pri vlastnom nahrávaní podstránky, po doplnení HTML a CSS kódu, je potrebné manuálne vložiť aj JavaSkript. Aktualizovať HTML a CSS je pomerne jednoduché, stačí ich vložiť do elementov stránky a prehliadač ich automaticky zobrazí. Pri nahrávaní skriptu je to o niečo zložitejšie. Ten je nahrávaný tak, že sa do elementov stránky pridá skript element, ktorému sa priradí adresa s cestou, na ktorej sa skript nachádza. Prehliadač si ho potom stiahne sám. Pre jednoduchosť sa JavaScript podstránky volá rovnako ako jej HTML kód.

3.3.3 Navigácia na stránke

Po príchode na stránku dochádzkového systému sa zobrazí úvodná obrazovka s prihlasovacími údajmi, emailom a heslom. Celá stránka je v anglickom jazyku. Po vyplnení prihlasovacích údajov, ktoré patria účtu administrátora sa zobrazí úvodná stránka. Na ľavej strane stránky je obrázok užívateľa s jeho menom, pod ktorým je záložkové menu. Toto menu umožňuje užívateľovi prepínať medzi jednotlivými dielčími podstránkami. Dielčie podstránky sú prehľad, zamestnanci, pridať užívateľa a nastavenia. V prehľade je tabuľka s menami pracovníkov, ktorý sú v práci. V záložke zamestnanci je zoznam zamestnancov s ich informáciami. Tento zoznam je možné zoraďovať abecedne, alebo číselne, pre každý druh informácie v tabuľke. Po rozkliknutí zamestnanca sa zobrazia jeho osobné údaje, dochádzka za aktuálny deň a nastavenia jeho profilu s možnosťou pridať záznam s minulým dátumom, ak ho zamestnanec zabudol vytvoriť pri odchode alebo príchode na pracovisko. Takto vytvorený záznam je odlíšený od ostatných skratkou (PC). Osobné údaje je možné meniť. Pre rýchlejšie hľadanie minulej dochádzky je možné zobraziť kalendár a listovať späť po jednotlivých mesiacoch. Jednotlivé záznamy je možné filtrom označiť červenou farbou a to nastavením neskorého príchodu, alebo minima odpracovaných hodín. Chýbajúce záznamy v dochádzke sa označujú čerenou automaticky. Dochádzku je možné stiahnuť v mesačnom rozsahu vo formáte CSV. V nastaveniach je možné zamestnanca z databázy vymazať. Pridať zamestnanca je záložka, v ktorej sa pridáva nový zamestnanec. Pre jeho pridanie je potrebné vyplniť všetky polia, ktoré sú meno, priezvisko, email, číslo jeho kartičky a funkciu administrátora. Ak server potvrdí, že email a číslo kartičky je ešte neobsadené vytvorí sa nový účet zamestnanca a na zadaný email príde odkaz pre vytvorenie nového hesla, ktoré si užívateľ vytvorí sám. Po zadaní hesla sa s ním môže prihlásiť do dochádzkového systému, ak jeho vytvorený účet mal administrátorské oprávnenie. V poslednej záložke nastavenie je možnosť zmeniť si profilový obrázok za jeden z predvolených.

4 Testovanie

Kvôli odhaleniu chýb a praktickému odskúšaniu bol dochádzkový systém testovaný firmou so štyrmi zamestnancami. Vo firme bol v prevádzke jeden pracovný týždeň. Majiteľ firmy ho následne písomne zhodnotil.

„Dochádzkový systém bol pre našu firmu prínosom, pretože doteraz sa zamestnanci zapisovali do zošita. Hneď po 2 dňoch zaznamenal neskoré príchody niektorých zamestnancov, avšak mal aj nedostatky. Server bol spustený na počítači, ktorý musel byť stále zapnutý v priestoroch firmy a menu bolo iba v anglickom jazyku bez možnosti zmeny. Pre zvýšenie kvality navrhujem namiesto čítačky kariet použiť otlačok prsta a prachotesné prevedenie obalu.“ (Ľubomír, majiteľ firmy, 20.5.2019)

Pri testovaní bol v prvý deň zistený závažný problém s ukončením serverového programu NodeJS kvôli neaktivite. Tento problém bol odstránený spusteným NodeJs cez npm program s názvom forever. Ten sa postará o to, aby sa NodeJS pri neaktivite neukončil.

Záver

Cieľom tejto práce bolo zhotoviť elektronický dochádzkový systém a tento cieľ sa podarilo splniť. Systém po priložení karty k fyzickej časti vytvára záznamy, ktoré sa cez server ukladajú do databázy. Príchod a odchod s poznámkou sa volí tlačidlami na zariadení, pričom predvolene je dopoludnia nastavený príchod a popoludní nastavený odchod. Tieto informácie sú zobrazené spolu s časom na LCD displeji, ktorý zobrazuje aj priebeh vytvorenia záznamu. V prípade neúspešného vytvorenia záznamu sa dôvod zobrazí. Menu je v jazyku angličtina. V zariadení sa čas pravidelne aktualizuje zo servera a udržiava sa v module skutočného času. Na server sa dochádzka odosiela prostredníctvom WiFi. Po prihlásení sa na webovú stránku servera, je možné dochádzku prehliadať. Kalendárne zobrazenie dochádzky je vhodné pre jej rýchlejšie listovanie. Stránka komunikuje len cez zabezpečené pripojenie HTTPS. Aj keď je server aktívny aj na porte 80 pre HTTP, je to iba kvôli presmerovaniu na HTTPS. Každý užívateľ pri prihlásení obdrží unikátny token, ktorý ho oddeľuje od ostatných prihlásených užívateľov, čím sa úroveň zabezpečenia ešte zvyšuje. Záznamy o dochádzke jednotlivých zamestnancov je možné prehliadať, upravovať a sťahovať vo formáte CSV, zároveň je možné odlíšiť ich červenou farbou, kvôli neskorému príchodu, alebo nedostatočnému počtu odpracovaných hodín. Doplnené záznamy z počítača sú odlíšené skratkou (PC). Pri nedostupnosti servera, alebo databázy, sú záznamy odoslané zo zariadenia zálohované na SD karte v jeho vnútri. Všetok kód je zálohovaný na stránke GitHub, kde je verejne dostupný[11], okrem serverových konštánt ako prihlasovacie údaje a overovacie reťazce. Maximálny počet zamestnancov nie je programovo obmedzený, avšak obmedzenie udáva maximálna využiteľná pamäť dostupná pre prehliadač. Pre malé firmy do 50 zamestnancov to postačuje. Kvôli odhaleniu chýb bol dochádzkový systém odskúšaný firmou so štyrmi zamestnancami na jeden pracovný týždeň.

Zoznam skratiek a symbolov

0x27	hexadecimálne číslo
CLK	hodinový signál (clock)
CPU	procesor (central processing unit)
ESP32	rodina mikrokontrolérov
I^2C	komunikačné rozhranie na prenos všeobecných dát
I^2S	komunikačné rozhranie na prenos zvuku
JSON	formát súboru (JavaScript Object Notation)
MISO	pin SPI zbernice (master in slave out)
MOSI	pin SPI zbernice (master out slave in)
npm	databáza modulov pre program NodeJS (node package manager)
pin	vývod (nožička) čipu
PLA	plast (Polylactic acid)
ppm	označenie jednej milióntiny (parts per milion)
QFN	puzdro čipu (Quad Flat No-leads package)
ROM	pamäť iba na čítanie (read only memory)
RTC	čítač skutočného času (real time counter)
Rx	pin UART zbernice na príjem dát (receive)
SCLK	hodinový signál (signal clock)
SD	úložné zariadenie (storage device)
SDA	linka I^2C na obojsmerný prenos dát
SPI	komunikačný rozhranie (Serial Peripheral Interfac)
SQL	jazyk pre prácu s databázou (Structured Query Language)
SRAM	pamäť (Static Random Access Memory)
SS(CS)	pin SPI zbernice (chip select / slave select)
TTL	druh komunikačnej logiky (Transistor–transistor logic)
Tx	pin UART zbernice na odosielanie dát (receive)
UART	komunikačné rozhranie na prenos dát

Zoznam použitej literatúry

[1] © Espressif Systems .: *ESP32 Series Datasheet.* [online]. Výrobca polovodičových čipov espressif.com, 2018. Version 2.7. Dostupné na internete dňa 9.12.2018: https://www.espressif.com/sites/default-/files/docume-ntation/esp32datasheet_en-.pdf

[2] HAMEDANI, Mosh.: *What is Node.js?.* [online]. databáza online videí: programming with Mosh, 2018. Dostupné na internete dňa 9.12.2018: https://www.youtube.com-/watch?v=uV-wtVBpw7RQ

[3] © Node.js Foundation.: *About Node.js®.* [online]. Stránka spoločnosti Joyent, Inc, 2018. Dostupné na internete dňa 9.12.2018: https://nodejs.org/en/about/

[4] MEAD, Andrew.: *The Complete Node.js Developer Course (2nd Edition):What is node.* [online]. Platené online kurzy udemy.com, 2018. Dostupné na internete dňa 9.12.2018: https://www.ude-my.com/the-complete-nodejs-developer-course-2learn-/v4/t/lecture /55 25226

[5] MEAD, Andrew.: *The Complete Node.js Developer Course (2nd Edition): The ObjectId.* [online]. Platené online kurzy udemy.com, 2018. Dostupné na internete dňa 9.12.2018: https://www.udemy.com/the-complete-nodejs-developer-course2/le-arn-/v4-/t/lecture/5677856

[6] Neznámy autor.: *Úvod do JSON.* [online]. Dostupné na internete dňa 9.12.2018: http://json.org/json-cz.html

[7] © MongoDB.: *What in no NoSQL.* [online]. Spoločnosť MongoDB, Inc, 2018. Dostupné na internete dňa 9.12.2018: https://www.mongodb.com/nosql-explained?jmp=footer

[8] TIŠNOVSKÝ, Pavel.: *Externí sériové sběrnice SPI a I²C.* [online]. Online časopis root.cz, 2008. Dostupné na internete dňa 9.12.2018: https://www.root.cz/clanky-/externi-seriove-sbernice-spi-a-i2c/ ISSN 1212-8309

[9] PEFHANY, Spehro.: *How does a bidirectional level shifter work*?. [online]. Internetové fórum stackexchange.com, 2015. Dostupné na internete dňa 9.12.2018: https://electronics.stackexchange.com/questions/173297/how-does-a-bidirection-al-level-shifter-work

[10] Dallas Semiconductor ®.: *DS3231 Datasheet.* [online]. Databáza technických listov alldatasheet.com, 2005. Dostupné na internete dňa 9.12.2018: http://html.all-datasheet.com/html-pdf/112132/DALLAS/DS3231/436/2 /DS3231.html

[11] VALENT, A.: *web-server*. [online]. Server správy verzií © GitHub, Inc., 2018. Dostupné na internete dňa 15.5.2019: https://github.com/andz885/web-server

[12] ABABEI, Cristinel.: *Lecture 12: SPI and SD cards*. [online]. Electrical Engineering Department, University at Buffalo, 2013. Dostupné na internete dňa 15.5.2019: http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf

[13] © MongoDB.: *MongoDB Download Center*. [online]. Spoločnosť MongoDB, Inc, 2018. Dostupné na internete dňa 15.5.2019: https://www.mongodb.com/download-center

[14] BOGENSBERGER, B. CEO, SCHLUETER, I. Chief Product Officer, VOSS, L. Chief Data Officer, UMLAH, D. Chief Operating Officer.: *About npm*. [online]. Databáza knižníc pre NodeJS, 2009. Dostupné na internete dňa 15.5.2019: https://www.npmjs.com/about

[15] Advanced Monolithic Systems.: *AMS1117 datasheet*. [online]. Výrobca polovodičových čipov Advanced Monolithic Systems, Inc., 2007. Dostupné na internete dňa 15.5.2019: http://www.advanced-monolithic.com/pdf/ds1117.pdf

[16] NXP.: *MFRC522 datasheet*. [online]. Výrobca polovodičových čipov NXP Semiconductors, 2016. Dostupné na internete dňa 15.5.2019: https://www.nxp.com-/docs/en/data-sheet/MFRC522.pdf